/\*Begum Zubeda\*/

Select \* from bank\_customer;

Select \* from bank\_account\_details;

Select \* from bank\_account\_relationship\_details;

Select \* from bank\_account\_transaction;

Select \* from bank\_customer\_export;

Select \* from bank\_customer\_messages;

Select \* from bank\_interest\_rate;

Select \* from bank\_inventory\_pricing;

Select \* from bank\_branch\_pl;

Select \* from department\_details;

Select \* from employee\_details;

use PGA02;

/\*Q1. Print product, price, sum of quantity more than 5 sold during all three months. \*/

Select Product, SUM(Quantity) as SumQuantity from bank\_inventory\_pricing

group by Product

having SumQuantity > 5;

/\*Q2. Print product, quantity , month and count of records for which estimated\_sale\_price is less than purchase\_cost\*/

Select Product, Quantity, Month from bank\_inventory\_pricing

where Estimated\_sale\_price < Purchase\_cost;

/\*Q3. Extarct the 3rd highest value of column Estimated\_sale\_price from bank\_inventory\_pricing dataset \*/

Select Estimated\_sale\_price from bank\_inventory\_pricing

order by Estimated\_sale\_price desc limit 2,1;

Select Estimated\_sale\_price from (

Select Estimated\_sale\_price from bank\_inventory\_pricing

order by Estimated\_sale\_price desc limit 3

) as temp order by Estimated\_sale\_price limit 1;

/\*Q4. Count all duplicate values of column Product from table bank\_inventory\_pricing\*/

Select Product, count(Product) NumberProducts from bank\_inventory\_pricing

group by Product

having count(Product) > 1;

/\*Q5. Create a view 'bank\_details' for the product 'PayPoints' and Quantity is greater than 2 \*/

Create or Replace view bank\_details

AS

Select \* from bank\_inventory\_pricing where Product = 'PayPoints' and Quantity > 2;

Select \* from bank\_details;

/\*Q6. Update view bank\_details1 and add new record in bank\_details1.

-- --example(Producct=PayPoints, Quantity=3, Price=410.67) \*/

Insert into bank\_details(Product, Quantity, Price) values('PayPoints', 3, 410.67);

Select \* from bank\_details;

/\*Q7. Real Profit = revenue - cost Find for which products, branch level real profit is more than the estimated\_profit in Bank\_branch\_PL. \*/

Select Branch, Product, (Revenue-Cost) as RealProfit, Estimated\_profit from bank\_branch\_pl

where (Revenue-Cost) > Estimated\_profit;

Create table Test as Select \*, (Revenue-Cost) as Real\_profit

from bank\_branch\_pl;

Select \* from Test where Real\_profit > Estimated\_profit;

/\*Q.8. Find the least calculated profit earned during all 3 periods \*/

Select Month, min(Revenue-Cost) as Min\_Profit from bank\_branch\_pl

group by Month;

/\*Q.9 In Bank\_Inventory\_pricing,

-- a) convert Quantity data type from numeric to character

-- b) Add then, add zeros before the Quantity field. \*/

Select Product, cast(Quantity as CHAR(10)) as Quantity from bank\_inventory\_pricing;

Select Product, concat('00', cast( Quantity as CHAR(10)) ) newQuantity from bank\_inventory\_pricing;

/\*Q10. Write a MySQL Query to print first\_name , last\_name of the titanic\_ds whose first\_name Contains ‘U’ \*/

Select FIRST\_NAME, LAST\_NAME from hsales where FIRST\_NAME LIKE '%u%';

/\*Q11. Reduce 30% of the cost for all the products and print the products whose calculated profit at branch is exceeding estimated\_profit . \*/

Select Product, (Revenue - (Cost - Cost\*.3)) as RealProfit, Estimated\_profit from bank\_branch\_pl

where (Revenue - (Cost - Cost\*.3)) > Estimated\_profit;

Select Product, (Revenue - (Cost\*.7)) as RealProfit, Estimated\_profit from bank\_branch\_pl

where (Revenue - (Cost\*.7)) > Estimated\_profit;

/\*Q12. Write a MySQL query to print the observations from the Bank\_Inventory\_pricing table excluding the values “BusiCard” And “SuperSave” from the column Product \*/

Select \* from bank\_inventory\_pricing where Product NOT IN("BusiCard", "SuperSave");

/\*Q13. Extract all the columns from Bank\_Inventory\_pricing where price between 220 and 300 \*/

Select \* from bank\_inventory\_pricing where Price BETWEEN 220 and 300;

/\*Q14. Display all the non duplicate fields in the Product from Bank\_Inventory\_pricing table and display first 5 records.\*/

Select DISTINCT Product from bank\_inventory\_pricing limit 5;

/\* Q15.Update price column of Bank\_Inventory\_pricing with an increase of 15% when the quantity is more than 3. \*/

Update bank\_inventory\_pricing set Price = Price + (Price\*.15) where Quantity > 3;

/\*Q16. Show Round off values of the price without displaying decimal scale from Bank\_Inventory\_pricing \*/

Select \*, round(Price, 0) Round\_Price from bank\_inventory\_pricing; /\* Price, how many decimal places\*/

/\*Q17.Increase the length of Product size by 30 characters from Bank\_Inventory\_pricing. \*/

Describe bank\_inventory\_pricing;

Alter table bank\_inventory\_pricing MODIFY Product CHAR(30);

/\*Q18. Add '100' in column price where quantity is greater than 3 and dsiplay that column as 'new\_price' \*/

Select \*, (Price + 100) new\_price from bank\_inventory\_pricing where Quantity > 3;

/\*Q19. Display all saving account holders have “Add-on Credit Cards" and “Credit cards" and Account\_type \*/

Select Customer\_id from bank\_account\_details where Account\_type = "SAVINGS" and Customer\_id IN (

Select Customer\_id from bank\_account\_details where Account\_type = "Credit Card" and Customer\_id IN

(Select Customer\_id from bank\_account\_details where Account\_type = "Add-on Credit Card")

);

/\*Q20.

# a) Display records of All Accounts , their Account\_types, the transaction amount.

# b) Along with first step, Display other columns with corresponding linking account number, account types

# c) After retrieving all records of accounts and their linked accounts, display the transaction amount of accounts appeared in another column.\*/

Select t.Account\_Number, r.Linking\_Account\_Number, r.Account\_type, t.Transaction\_Amount

from bank\_account\_transaction as t

Inner Join bank\_account\_relationship\_details as r

ON t.Account\_Number = r.Account\_Number;

/\*Q21. Display all type of “Credit cards” accounts including linked “Add-on Credit Cards"

# type accounts with their respective aggregate sum of transaction amount.

# Ref: Check linking relationship in bank\_transaction\_relationship\_details.

# Check transaction\_amount in bank\_account\_transaction.\*/

Select sum(t.Transaction\_amount) as Sum\_Transaction, t.Account\_Number, r.Linking\_Account\_Number, r.Account\_type

from bank\_account\_transaction t INNER JOIN bank\_account\_relationship\_details as r

ON t.Account\_Number = r.Account\_Number

where r.Account\_type IN("Add-on Credit Card", "Credit Card")

group by t.Account\_Number;

Select sum(t.Transaction\_amount) as Sum\_Transaction, t.Account\_Number, r.Linking\_Account\_Number, r.Account\_type

from bank\_account\_transaction t, bank\_account\_relationship\_details as r

where r.Account\_type IN("Add-on Credit Card", "Credit Card") and t.Account\_Number = r.Account\_Number

group by t.Account\_Number;

/\*Insert into bank\_account\_relationship\_details values(NULL,'9000-1700-7777-4321',"Credit Card",'5000-1700-9800'),

(NULL,'5900-1900-9877-5543',"Add-on Credit Card",'9000-1700-7777-4321'),

(NULL,'5800-1700-9800-7755',"Credit Card",'4000-1956-5698'),

(NULL,'5890-1970-7706-8912',"Add-on Credit Card",'5800-1700-9800-7755');\*/

/\*Q.22 Compare the aggregate transaction amount of current month versus aggregate transaction with previous months.

# Display account\_number, transaction\_amount ,

-- sum of current month transaction amount ,

-- current month transaction date ,

-- sum of previous month transaction amount ,

-- previous month transaction date.\*/

Select avg(Transaction\_amount) as AVG\_Transaction, sum(Transaction\_amount) as Sum\_Transaction, month(Transaction\_date) as Transaction\_month from bank\_account\_transaction

group by month(Transaction\_date)

order by Transaction\_date desc;

/\*Q23. Display individual accounts absolute transaction of every next month is greater than the previous months .\*/

Select t1.Account\_Number, abs(t1.Transaction\_amount) as Abs\_Transaction\_amount, t1.Transaction\_date from bank\_account\_transaction t1

where t1.Account\_Number IN (

Select t2.Account\_Number from bank\_account\_transaction t2

where month(t1.Transaction\_date) < month(t2.Transaction\_date) and abs(t1.Transaction\_amount) > abs(t2.Transaction\_amount)

)

order by t1.Transaction\_date desc;

/\*Q24. Find the no. of transactions of credit cards including add-on Credit Cards \*/

Select count(\*) as Num\_Transaction from bank\_account\_transaction where Account\_Number IN(

Select Account\_Number from bank\_account\_relationship\_details where Account\_type IN("Credit Card", "Add-on Credit Card")

);

/\*Q25. From employee\_details retrieve only employee\_id , first\_name ,last\_name phone\_number ,salary, job\_id where department\_name is Contracting (Note

#Department\_id of employee\_details table must be other than the list within IN operator. \*/

Select e.EMPLOYEE\_ID, e.FIRST\_NAME, e.LAST\_NAME, e.PHONE\_NUMBER, e.SALARY, e.JOB\_ID, d.DEPARTMENT\_NAME from employee\_details e

Inner Join department\_details d

ON e.DEPARTMENT\_ID = d.DEPARTMENT\_ID;

/\*Q26. Display savings accounts and its corresponding Recurring deposits transactions are more than 4 times.\*/

Select a.Account\_Number, a.Account\_type, t.Transaction\_amount, t.Transaction\_date from bank\_account\_details a

Inner Join bank\_account\_transaction t ON a.Account\_Number = t.Account\_Number

where t.Account\_Number IN (

Select Account\_Number from bank\_account\_transaction

where a.Account\_type IN ('SAVINGS', "RECURRING DEPOSITS")

group by Account\_Number

having count(Account\_Number) > 3

);

/\*Q27. From employee\_details fetch only employee\_id, ,first\_name, last\_name , phone\_number ,email, job\_id where job\_id should not be IT\_PROG.\*/

Select EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, PHONE\_NUMBER, JOB\_ID from employee\_details where JOB\_ID <> 'IT\_PROG';

/\*Q29. From employee\_details retrieve only employee\_id , first\_name ,last\_name phone\_number ,salary, job\_id where manager\_id is '60' (Note

#Department\_id of employee\_details table must be other than the list within IN operator.\*/

Select EMPLOYEE\_ID, FIRST\_NAME, LAST\_NAME, PHONE\_NUMBER, SALARY, JOB\_ID from employee\_details

where MANAGER\_ID = 60;

Select e.EMPLOYEE\_ID, e.FIRST\_NAME, e.LAST\_NAME, e.PHONE\_NUMBER, e.SALARY, e.JOB\_ID, d.MANAGER\_ID

from employee\_details e

Inner Join department\_details d

ON e.EMPLOYEE\_ID = d.EMPLOYEE\_ID

where d.MANAGER\_ID = 60;

/\*Q30. Create a new table as emp\_dept and insert the result obtained after performing inner join on the two tables employee\_details and department\_details.\*/

Create table emp\_dept

AS

Select e.\*, d.DEPARTMENT\_NAME, d.LOCATION\_ID from employee\_details e

Inner Join department\_details d

ON e.EMPLOYEE\_ID = d.EMPLOYEE\_ID;

Select \* from emp\_dept;